#2 cc 3/18/02

520.41151X00

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant(s): H. NISHIYAMA

Serial No.:

Filed: February 7, 2002

Title: DATA PREFETCH METHOD FOR INDIRECT REFERENCES

Group:

LETTER CLAIMING RIGHT OF PRIORITY

Honorable Commissioner of
Patents and Trademarks
Washington, D.C. 20231

February 7, 2002

Sir:

Under the provisions of 35 USC 119 and 37 CFR 1.55, the applicant(s) hereby claim(s) the right of priority based on Japanese Patent Application No.(s) 2001-299718 filed September 28, 2001.

A certified copy of said Japanese Application is attached.

Respectfully submitted,

ANTONELLI, TERRY, STOUT & KRAUS, LLP

Carl I. Brundidge
Registration No. 29,621

CIB/nac
Attachment
(703) 312-6600

n70574 us

日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　９月２８日

出願番号
Application Number: 特願２００１－２９９７１８

出願人
Applicant(s): 株式会社日立製作所

２００１年１２月２１日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３１０９９７６

【書類名】　特許願
【整理番号】　NT01P0422
【提出日】　平成13年 9月28日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F　9/45
【発明者】
　【住所又は居所】　神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所　システム開発研究所内
　【氏名】　西山　博泰
【特許出願人】
　【識別番号】　000005108
　【氏名又は名称】　株式会社日立製作所
【代理人】
　【識別番号】　100068504
　【弁理士】
　【氏名又は名称】　小川　勝男
　【電話番号】　03-3661-0071
【選任した代理人】
　【識別番号】　100086656
　【弁理士】
　【氏名又は名称】　田中　恭助
　【電話番号】　03-3661-0071
【選任した代理人】
　【識別番号】　100094352
　【弁理士】
　【氏名又は名称】　佐々木　孝
　【電話番号】　03-3661-0071
【手数料の表示】
　【予納台帳番号】　081423

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　間接参照データプリフェッチ方法

【特許請求の範囲】

【請求項１】

プリフェッチ命令を持つマイクロプロセッサに対する命令を生成するプログラミング言語のコンパイラにおいて、配列の間接参照を行なうようなプログラムに対して、添字として使われる間接参照の配列の値が、一定の規則に従って増減すること、および、その増加率あるいは減少率が一定範囲内であるとき、キャッシュラインに対応して間接参照のプリフェッチの命令を作成するデータプリフェッチ方法。

【請求項２】

前記添字として使われる間接参照の配列の値の増減、および、その増加率あるいは減少率の判定がソースコード上に記述されたディレクティブの解析により行われる請求項１記載のデータプリフェッチ方法。

【請求項３】

前記添字として使われる間接参照の配列の値の増減、および、その増加率あるいは減少率の判定がコンパイラに対して指定されたオプションの解析により行われる請求項１記載のデータプリフェッチ方法。

【請求項４】

前記添字として使われる間接参照の配列の値の増減、および、その増加率あるいは減少率の判定が添字配列の要素の定義式の解析により行われる請求項１記載のデータプリフェッチ方法。

【請求項５】

前記添字として使われる間接参照の配列の値の増減、および、その増加率あるいは減少率の判定が間接参照を含むループのソースコードをユーザに提示して、各添字に関してその変化量をユーザが指示することにより行われる請求項１記載のデータプリフェッチ方法。

【請求項６】

他の命令の実行と並行して主記憶からキャッシュメモリにデータを転送するプ

リフェッチ命令を有する計算機上で実行されるプログラムをコンパイルするコンパイラであって、

（ａ）プログラム中のループのソースプログラムを中間語へ変換する処理、

（ｂ）前記変換された中間語に基づいて、前記ループのループ構造を認識して第２の中間語とループ表を生成する処理、

（ｃ）前記生成された第２の中間語とループ表を参照してループに関して解析を行ないループ中の間接参照を認識し第３の中間語を生成する処理する処理、

（ｄ）前記生成された第３の中間語とループ表を参照して、認識した間接参照に関するプリフェッチ命令を生成する処理、

よりなることを特徴とするコンパイラ。

【請求項７】

（ａ）プログラム中のループのソースプログラムを中間語へ変換する処理のためのプログラム、

（ｂ）前記変換された中間語に基づいて、前記ループのループ構造を認識して第２の中間語とループ表を生成する処理のためのプログラム、

（ｃ）前記生成された第２の中間語とループ表を参照してループに関して解析を行ないループ中の間接参照を認識し第３の中間語を生成する処理する処理のためのプログラム、

（ｄ）前記生成された第３の中間語とループ表を参照して、認識した間接参照に関するプリフェッチ命令を生成する処理のためのプログラム、

を格納した記憶媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はデータプリフェッチ方式に関し、さらに詳しくは、プリフェッチ命令を備えるプロセッサ向けに、プリフェッチを効果的に利用することによって間接配列参照を高速化するコンパイル方法に関する。

【０００２】

【従来の技術】

命令レベル並列度や周波数の向上によってマイクロプロセッサの性能は飛躍的に向上を続けている。これに対して、計算機の主記憶を構成するＤＲＡＭの性能向上はプロセッサ性能の向上に比べ低いレベルに留まっている。この結果、主記憶参照に要するサイクル数は増加を続ける傾向にある。

【０００３】

図２に示すように、多くのマイクロプロセッサでは、このような主記憶参照時間を隠蔽するための方式として、主記憶と比較して高速に参照可能なキャッシュと呼ばれる少容量のメモリをプロセッサと主記憶の間に配置し、最近参照したデータをキャッシュ上に置くことによって、主記憶参照に要する時間を短縮するという方式を採っている。ただし、これだけでは、キャッシュ上に無いデータを参照した場合に待ちが生じるので、主記憶からキャッシュへ指定したアドレスのデータを、他の命令の実行と並行して先行的に転送するデータプリフェッチ命令を用意している。

【０００４】

コンパイラはソースプログラムを解析して、主記憶参照サイクルが隠蔽できるようにプリフェッチ命令を生成するコードの最適化を行なう。ここで、演算に利用するデータ長は一般に１～８バイト程度の大きさである。また、主記憶からキャッシュへのデータ転送は１２８バイトや６４バイトの大きさのキャッシュラインと呼ばれる単位毎に行なわれる。従って、一度のデータプリフェッチによって複数回のループ繰返しで参照するデータをキャッシュ上に転送することができる。

【０００５】

文献「T.C.Mowry 他、Design and Evaluation of a Compiler Algorithm for Prefetching, Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems, pp.62-73, 1992」に示されているような従来技術では、ループ中のデータ参照の空間的局所性あるいは時間的局所性をループ中の配列添字の解析から求め、転送したキャッシュラインの再利用率が一定以上の場合にプリフェッチを行なう。この際、ループ展開と組み合わせることにより、同一キャッシュラインへの冗長なプリフ

ェッチを行なわないようにし、プリフェッチに伴う命令オーバヘッドを低減している。

【０００６】

図３を用いて従来技術を説明する。

【０００７】

図３（ａ）は本願発明を適用して効果のある、間接参照を伴うループ繰返し処理を行なうソースコードの例を示す。このコードは、配列Ｌ［．．．］の値を添え字として、配列Ａ［．．．］の値の総和を求める処理をｉからＮまで実行せよ、ということを意味する。このような間接参照コードは、疎行列の処理などで頻繁に利用されるものである。

【０００８】

図３（ｂ）は、このソースコードを従来技術で最適化するとともにプリフェッチ命令を挿入したコードの例を示す。従来技術では、コンパイル時に配列添字を解析し、キャッシュラインの再利用率の高い配列参照のみをプリフェッチ対象とするので、連続参照される配列Ｌ［．．．］のみがプリフェッチの対象となり、再利用率の不明な配列Ａ［．．．］についてはプリフェッチの対象となっていない。なお、ここでαはプリフェッチによって主記憶からキャッシュへデータが到着するまでのループ繰返し回数を表す。また、この例では、プロセッサのキャッシュラインの大きさは３２バイトとする。演算に利用するデータ長がたとえば８バイトの大きさである場合は、４要素のデータが主記憶から一括して転送される。従って、この例では、４回の繰返しに一度プリフェッチを行なうようにコード化されている。

【０００９】

従来方式でも、キャッシュラインの再利用率が不明な配列Ａ［．．．］についても、常にプリフェッチを発行するようにすることでプリフェッチの適用対象とすることは可能である。但し、その場合には、１つの間接参照に対して１つのプリフェッチを発行する必要が生じ命令オーバヘッドが増大する。このようにして最適化されたコードの例を図３（ｃ）に示す。図３（ｃ）に示すコードを図３（ｂ）のコードと比較すると、プリフェッチのための添え字参照と配列Ａ［．．．

］のプリフェッチのための命令が増加するために、命令数が多くなっている。このため、一般には間接参照に対するプリフェッチは行なわれていない。

【００１０】

【発明が解決しようとする課題】

従来技術による最適化コードでは、間接参照データのプリフェッチのための命令オーバヘッドを削減しながら、間接参照される配列の参照時におけるキャッシュミスサイクルを隠蔽することができないので、プログラムの実行性能が低下してしまうという問題が生じる。

【００１１】

【課題を解決するための手段】

上記、従来技術において、間接参照配列に対するプリフェッチのためのオーバヘッドが増加するのは、添字配列の要素の値が離散的であることを、プログラム解析の結果、コンパイラが想定しているためである。すなわち、添字配列の要素の値が離散的であると、再利用率が悪くなるからである。添字配列の要素は任意の値をとり得るので、不正な最適化を行なわないためにはこのように仮定する必要がある。

【００１２】

しかしながら、実際のプログラムでは、添字配列の要素の値は、完全に離散的な値ではなく、例えば｛１、２、３、４、６、７、８、９、５１、５２、５３、５４、５６．．．｝のように、局所的に見るとほぼ連続的な値をとることが多い。プリフェッチ命令は主記憶からキャッシュメモリへデータを移動するだけであり、キャッシュへのデータ移動はアプリケーションプログラムから見た場合には、データをロードする際のサイクル数のみが変化することとなる。

【００１３】

従って、上記の添字配列に関して、添字配列の値が連続的、あるいは、一定の規則に従って変化すると認識した結果と、プログラムが実際に動作する際に添字配列がとった値に一部相違があったとしても、プログラムの結果には違いは生じないことが分かる。

【００１４】

本発明では、このように配列添字が連続的あるいは、一定の規則にしたがって変化する可能性が高い場合を、コンパイラ解析、ユーザ指示などを元に認識することにより、従来技術において、間接参照データのプリフェッチのための命令オーバヘッドを削減しながら、間接参照される配列の参照時におけるキャッシュミスサイクルを隠蔽することができない、という問題を解決する。

【００１５】

具体的には、まず、コンパイラ解析、ユーザ指示などにより、添字配列の値が一定の増分値を持つことを認識する。この結果、当該添字配列を添字として参照される配列要素が一定のアドレス間隔で参照されることがわかる。よって、ループ展開などの従来技術の最適化と組み合わせて、プリフェッチを一定のループ繰返し毎に発行するようにする。この結果、同一キャッシュラインを参照する間接参照配列に対するプリフェッチの命令数の増加を抑止することが可能となる。

【００１６】

以上により、プリフェッチのための命令数のオーバヘッドの増加を低く保って、間接参照配列に対するプリフェッチを適用することが可能になり、プリフェッチ命令の実行によるメモリ参照サイクルの隠蔽効果と併せて、プログラムの実行性能を向上することができる。

【００１７】

【発明の実施の形態】

以下、本発明の１つの実施例を図を参照しながら説明する。

【００１８】

図２は本発明を実施する計算機システムの１つの例である。計算機システムはキャッシュメモリ２０２を含むマイクロプロセッサ２０１、主記憶２０３、ディスク２０４からなり、プログラムはディスク２０４に格納される。コンパイラおよびソースプログラムはディスク２０４に格納され、プロセッサ２０１に読み出されコンパイル処理を行なう。コンパイル結果のプログラムはディスク２０４に格納され、同様にプロセッサ２０１に読み出され実行される。プロセッサ２０１で通常のメモリ参照命令の実行を行なう場合には、まずキャッシュメモリ２０２に参照対象のデータがあるかどうかを調べ、キャッシュメモリ２０２にデータが

存在すればそのデータを参照し、キャッシュメモリ２０２に参照対象のデータが存在しなければ主記憶２０３上の当該データを参照すると共に、当該データの属するキャッシュラインのコピーをキャッシュメモリ２０２に置く。キャッシュメモリの参照は主記憶の参照に比べて高速であり、参照対象のデータがキャッシュメモリ上にあればメモリ参照によって発生する待ち時間を減少することができる。プリフェッチ命令は、他の命令の実行と同時に主記憶２０３からキャッシュメモリ２０２へ参照対象のデータが属するキャッシュラインを移動する命令であり、主記憶２０３からキャッシュメモリ２０２へキャッシュラインを移動するのに十分なサイクル数だけ前にプリフェッチ命令を発行しておけば、主記憶２０３からキャッシュメモリ２０２へのデータの転送を行なっている間に他の命令を実行することができるので、当該データを参照するための待ちは無くなる。

【００１９】

図１は本発明を実施するコンパイラの最適化処理フローの構成の１つの例を示す。図１において、実線は制御の流れを、破線はデータの流れを示している。なお、一般にコンパイラの最適化処理フローは様々な最適化処理から成るが、ここでは本発明に関連する処理のみを示している。

【００２０】

図１に示す実施例では、最適化処理処理１０１により中間語１０５に対して、間接参照に対するプリフェッチ命令を生成する最適化処理を適用し、中間語１０９を生成する。最適化処理１０１では、まずループ構造認識処理１０２によって中間語１０５を解析してプログラム中のループを認識し、中間語１０６とループ表１０８を生成する。次に、間接参照認識処理１０３では、ループ構造認識処理１０２で認識したループに関して解析を行ない、ループ中の間接参照を認識する。次に、間接参照プリフェッチ生成処理１０４では、認識した間接参照に関して、プリフェッチ命令を生成し、最適化された中間語１０９を生成する。これらの処理のうち、ループ構造認識処理１０２および間接参照認識処理１０３は、例えば「A.V.Aho他、Compilers: Principles、Techniques and Tools、Addison-Wesley、1986, pp.513-584, Chapter 9: Code Generation, pp. 585-722, Chapter 10: Code Optimization」などに開示されている従来技術により実施することができ

る。

【００２１】

本発明の特徴的な処理である、間接参照プリフェッチ生成処理１０４の処理フローを図４に示す。間接参照プリフェッチ生成処理１０４は、ステップ４０１で処理を開始し、ステップ４０２でプログラム中のループ集合を図１のループ表１０８を参照して変数Ｌに求める。次にステップ４０３では、集合Ｌが空集合であるか否かを確かめ、処理すべきループが存在しなければステップ４１０に制御を移して、処理を終了する。処理すべきループがある場合、Ｌから１要素取り出し変数Ｉに代入する。次に、ステップ４０５では、図１の間接参照認識処理１０３の結果を参照して、ループＩ中の間接参照の集合を変数Ｍに求める。次に、ステップ４０６では集合Ｍが空集合であるか否かを確かめる。Ｍが空集合であれば、ステップ４０３に制御を移し次のループを処理する。Ｍが空集合でなければ、ステップ４０７に制御を移し、集合Ｍから間接参照を１つ取り出して変数ｍに代入する。次にステップ４０８では、変数ｍに代入された間接参照を調べ、ｍの参照が連続であるか否かを確かめる。ｍの参照が連続であれば、プリフェッチによる主記憶参照サイクルの隠蔽効果が期待できるので、ステップ４０９に制御を移し、間接参照に対するプリフェッチを生成する。ｍの参照が連続でなければプリフェッチによる主記憶参照サイクルの隠蔽効果が期待できないので、ステップ４０６に制御を移して次の間接参照を処理する。ステップ４０９のプリフェッチ生成については、文献「V.Santhanam他、Data Prefetching on the HP PA-8000、In Proceedings of the 24th Annual International Symposium on Computer Architecture、pp.264-273、1997」に示されているような技術を利用して、同一キャッシュラインを参照するプリフェッチ命令の発行を削減して、プリフェッチを生成する。

【００２２】

ステップ４０８では、ループ中の間接参照について、当該参照が連続であるか否かを判定する。先にも述べたように、本発明は、実際のプログラムでは、添字配列の要素の値は、完全に離散的な値ではなく、例えば｛１、２、３、４、６、７、８、９、５１、５２、５３、５４、５６．．．｝のように、局所的に見ると

ほぼ連続的な値をとることが多いことに着目したものであるから、この例では、配列の要素の値が１から９までの部分および５１、５２、５３、５４、５６の部分では「連続である」と判定するものでなければならない。勿論、配列の要素の値が９から５１と飛ぶ部分では「不連続である」と判定するものでなければならない。

【００２３】

この判定に関しては、以下に示すような方法で実現するのが良い。

（１）ソースプログラムのコードを解析することにより自動判定する。

（２）コンパイラオプションによりユーザが指示する。

（３）コンパイラディレクティブによりユーザがソースコード上に指示する。

（４）提示されたソースコードに対して、ユーザが対話的に指示する。

【００２４】

図５は、上記（１）のコンパイラによる自動解析を行なう場合の処理フロー４０８を表している。図５の自動解析処理では、ステップ５０１で処理を開始し、解析対象となる配列をステップ５０２で変数ｍに格納する。次に、ステップ５０３でｍの定義式を求め、変数ｅに格納する。次にステップ５０４で式ｅが「ｉ＊α＋β」の形式の線形式であるか否かを確かめる。なお、ここで、ｉは帰納変数とする。線形式でない場合は、ステップ５０７に制御を移し、ｍは非連続であるものとし、ステップ５０８に制御を移して処理を終了する。ｅが線形式の場合は、ステップ５０５でｅの増分値｜α｜が一定値以下であるか否かを判定する。｜α｜が一定値以下であれば、ステップ５０６に制御を移してｍは連続であるものとし、ステップ５０８で処理を終了する。｜α｜が一定値以下でなければ、ステップ５０７に制御を移してｍは連続でないものとし、ステップ５０８で処理を終了する。例として、添字配列Ｌの定義ループが図８に示すものであるとする。ここで、Ｌの添字定義式は「ｉ＊２」であるので、添字の増分値は２であることがわかる。連続性の判定の閾値の判定値が４であるものとすると、Ｌを添字配列として利用する間接参照はプリフェッチ適用対象であると認識される。

【００２５】

図６は上記（２）のコンパイラオプションによる指示の例を示している。この

例では、添字配列Ｌの増分値が２であることをオプションにより指示している。コンパイラは、この増分値を元に、上記の自動解析の場合と同様に、プリフェッチの適用の可否を判定する。

【００２６】

同様に、図７は上記（３）のコンパイラディレクティブの使用例を示している。この例では、「*option increase(L,2)」というディレクティブにより、後続ループ中の添字配列Ｌの増分値が２であることを指示している。図５の例と同様に、この増分値を元に、プリフェッチの適用の可否を判定する。

【００２７】

図７のようなループに対するディレクティブ指示をユーザが与える場合、ユーザが直接ソースプログラム中にディレクティブを記述しても良いが、上記（４）のように、図１の間接参照認識処理１０３で認識した間接参照を含むループをコンピュータのディスプレー上に表示するものとして、各間接参照について添字の変化量の予測をユーザが対話的に指示することも可能である。

【００２８】

図９に、図３（ａ）の例に本発明を適用して最適化を実現した結果のコードを示す。添字配列の自動解析、あるいは、ディレクティブ等によるユーザ指示の解析の結果、間接参照Ａ［Ｌ［ｉ］］の添字配列Ｌの増分値が２であることがわかったものとする。キャッシュラインの大きさが３２バイト、参照するデータ長が４バイトであるとすると、ループを４倍展開して図９に示すように、元のソースコードの繰返し４回に一度配列ＬおよびＡに対するプリフェッチを行なうようにする。これにより、図３（ｃ）と対比しても明らかなように、プリフェッチによる命令オーバヘッドを低減して、間接参照に対するプリフェッチを行なうことができる。

【００２９】

【発明の効果】

本発明によれば、間接配列参照を行なうようなプログラムに関して、データプリフェッチを効果的に適用し、プログラムの実行性能を向上することができる。

【図面の簡単な説明】

【図１】

本発明を実施するコンパイラの最適化処理フローの構成の１つの例を示す図。

【図２】

本発明を実施する計算機システムの例を示すブロック図。

【図３】

（ａ）は本願発明を適用して効果のある、間接参照を伴うループ繰返し処理を行なうソースコードの例を、（ｂ）および（ｃ）は従来技術によりプリフェッチを導入して最適化したプログラムの例を、それぞれ示す図。

【図４】

本願発明による間接参照プリフェッチ処理の処理フローの例を示す図。

【図５】

本願発明によるループ中の間接参照が連続であるか否かを判断する処理フローの例を示す図。

【図６】

ループ中の間接参照が連続であるか否かを判断するためにコンパイラオプションにより添字配列Ｌの増分値を指示するためのオプションの例を示す図。

【図７】

ループ中の間接参照が連続であるか否かを判断するためにディレクティブにより添字配列Ｌの増分値を指示する例を示す図。

【図８】

添字配列定義コードの例を示す図。

【図９】

本願発明を適用してプリフェッチを導入して最適化したプログラムの例を示す図。

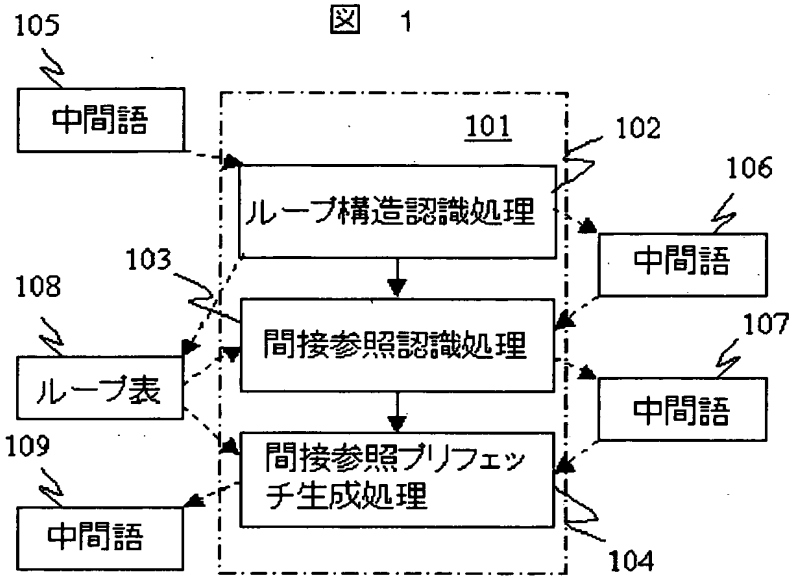
【符号の説明】

１０１…最適化処理フロー、１０２…ループ構造認識処理、１０３…間接参照認識処理、１０４…間接参照プリフェッチ生成処理、１０５，１０６，１０７，１０９…中間語、１０８…ループ表、２０１…マイクロプロセッサ、２０２…キャッシュメモリ、２０３…主記憶２０３、２０４…ディスク。

【書類名】　図面

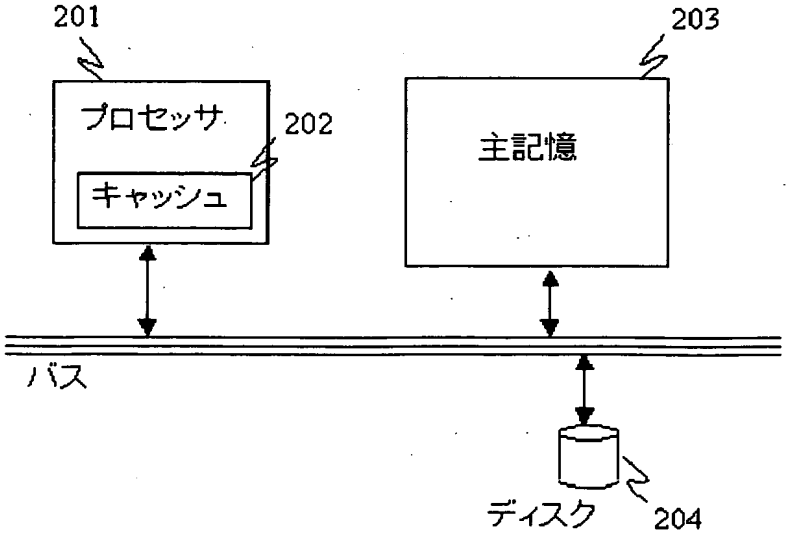
【図１】

図　1



【図２】

図　2



【図３】

図 3

(a)

```
DO i=1,N
  s=s+A[L[i]]
END DO
```

(b)

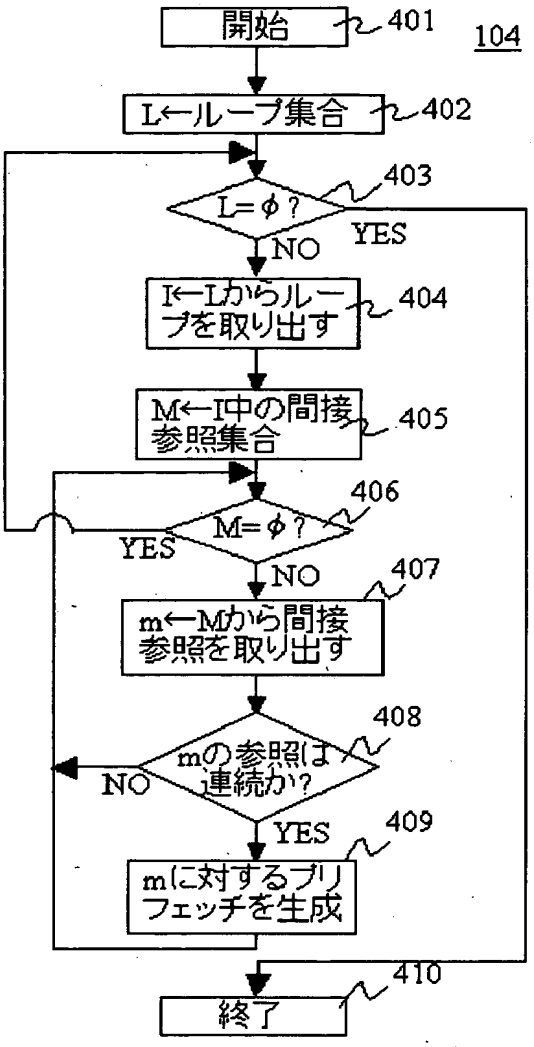
```
DO i=1,N,4
 PREFETCH L[i+α]
 s=s+A[L[i]]
 s=s+A[L[i+1]]
 s=s+A[L[i+2]]
 s=s+A[L[i+3]]
END DO
```

(c)

```
DO i=1,N,4
 PREFETCH L[i+2α]
 t0=L[i+α]
 PREFETCH A[t0]
 t1=L[i+1+α]
 PREFETCH A[t1]
 t2=L[i+2+α]
 PREFETCH A[t2]
 t3=L[i+3+α]
 PREFETCH A[t3]
 s=s+A[L[i]]
 s=s+A[L[i+1]]
 s=s+A[L[i+2]]
 s=s+A[L[i+3]]
END DO
```

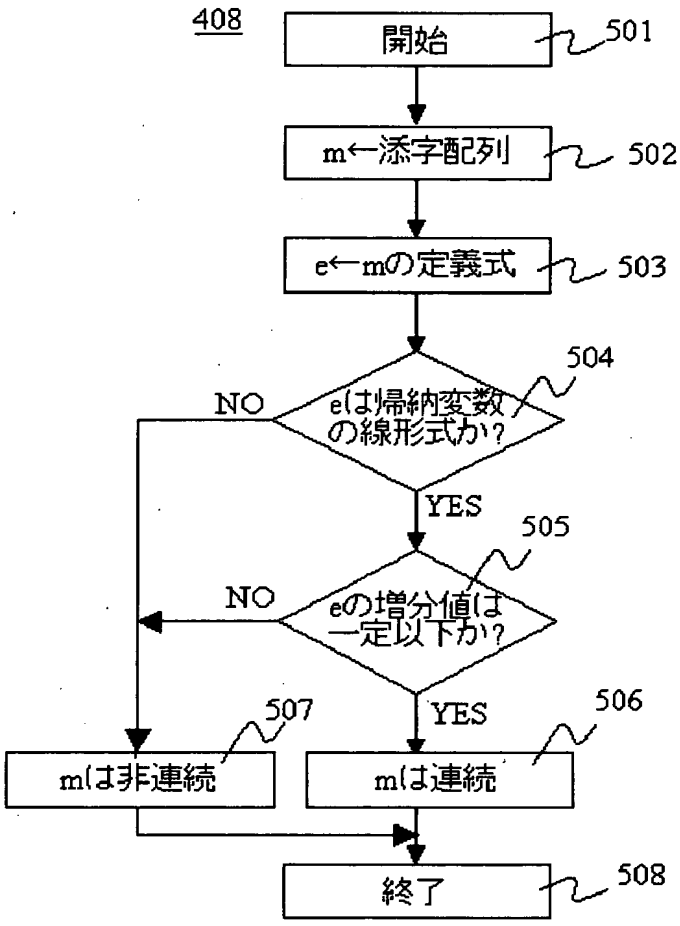
【図４】

図 4


開始
401
104
L←ループ集合
402
L=φ?
403
YES
NO
I←Lからループを取り出す
404
M←I中の間接参照集合
405
M=φ?
406
YES
NO
m←Mから間接参照を取り出す
407
mの参照は連続か?
408
NO
YES
mに対するプリフェッチを生成
409
終了
410

【図５】

図 5



【図６】

図 6

-array_index L=2

【図７】

図 7

```
 *option increase(L,2)
DO i=1,N
 s=s+A[L[i]]
END DO
```

【図８】

図 8

```
DO i=1,N
 L[i]=i*2
END DO
```

【図９】

図 9

```
DO i=1,N,4
 PREFETCH L[i+α]
 t0=L[i+α]
 PREFETCH A[t0]
 s=s+A[L[i]]
 s=s+A[L[i+1]]
 s=s+A[L[i+2]]
 s=s+A[L[i+3]]
END DO
```

【書類名】要約書

【要約】

【課題】　命令オーバヘッドを削減して、間接参照配列についてもデータプリフェッチを行なうことにより、間接配列参照を行なうプログラムの実行性能を向上すること。

【解決手段】　従来のコンパイラでは、はデータ参照の局所性が検出できなかったため、間接参照配列はデータプリフェッチの対象とならなかった。間接参照配列について、添字の値が、単調増加あるいは単調減少する傾向を持つことを、ユーザ指示やコンパイラによる解析によって検出し、この情報を利用することによって冗長なプリフェッチ命令の発行を抑止するとともに、間接参照配列についてもデータプリフェッチを行なう。

【選択図】　図1

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　　東京都千代田区神田駿河台４丁目６番地
氏　名　　株式会社日立製作所